# CRC

## Objective

Cyclic redundancy checks are a common method in communication systems - in particular in embedded systems - employed to assure that the data sent and received are the same. The references provided below  give us an overview and a guide to the theory as well as practical considerations.

A particular design then would have to make a choice based on the level of assurance it provides - in particular about the ability of the algorithm to detect single bit, multiple bit errors.

**NOTE**

The references provided here cover the theoretical underpinnings - thus guiding the selection of an appropriate CRC algorithm for each checksum size (the polynomial selection). It is not a goal of this projectlet to evaluate the polynomials themselves - instead relying on experts and published literature to choose an appropriate polynomial.

**REFERENCE**

http://www.ross.net/crc/download/crc_v3.txt
https://en.wikipedia.org/wiki/Computation_of_cyclic_redundancy_checks
http://www.sunshine2k.de/articles/coding/crc/understanding_crc.html

# User needs and requirements

This projectlet is to fulfill the need of software engineers.

| Id | Need/Requirement |
|---|---|
| 1 | The user needs a crc16 algorithm to calculate the crc of a block of data |
| 2 | Given a polynomial, the user needs to generate an initial table |
| 3 | The user needs to calculate the CRC of a given string - using the polynomial specified |
| 4 | The user needs to calculate the CRC of a file - using the polynomial specified |
| 5 | The user needs a way to experiment with noise ie by introducing errors in the message |

## Specifications

| Id | Wants |
|---|---|
| 1 | Use -p or —polynomial to specify the polynomial of 16 bits. The default algorithm will be used with no specification of the polynomial. |

| Id | Wants |
|---|---|
| 2 | The switch -x or —hex-string indicates that the argument is to be treated as a hex string and converted into binary to compute the CRC |
| 3 | The switch -f or —file indicates the argument is a file for which the CRC is requested. |
| 4 | The switch -l or —lines indicates the argument is a file and each line of this text file is treated as a null terminated string and the CRC values are computed. |
| 5 | If none of these switches is present, the argument is taken as a string. |
| 6 | The switch -t or —table just prints out the initial CRC table for the indicated polynomial |
| 7 | The switch -n or —noise directs the tool to introduce bit errors. The argument is the number of bits subject to errors. An attribute of the polynomial is the sensitivity to bit errors. |

# Example usage

#### DISPLAY THE DEFAULT CRC TABLE

```
dotnet bin/Debug/netcoreapp2.2/crc.dll -t
 0000 :  0000 c0c1 c181 0140 c301 03c0 0280 c241
 0001 :  c601 06c0 0780 c741 0500 c5c1 c481 0440
 0002 :  cc01 0cc0 0d80 cd41 0f00 cfc1 ce81 0e40
 0003 :  0a00 cac1 cb81 0b40 c901 09c0 0880 c841
 0004 :  d801 18c0 1980 d941 1b00 dbc1 da81 1a40
 0005 :  1e00 dec1 df81 1f40 dd01 1dc0 1c80 dc41
 0006 :  1400 d4c1 d581 1540 d701 17c0 1680 d641
 0007 :  d201 12c0 1380 d341 1100 d1c1 d081 1040
 0008 :  f001 30c0 3180 f141 3300 f3c1 f281 3240
 0009 :  3600 f6c1 f781 3740 f501 35c0 3480 f441
 0010 :  3c00 fcc1 fd81 3d40 ff01 3fc0 3e80 fe41
 0011 :  fa01 3ac0 3b80 fb41 3900 f9c1 f881 3840
 0012 :  2800 e8c1 e981 2940 eb01 2bc0 2a80 ea41
 0013 :  ee01 2ec0 2f80 ef41 2d00 edc1 ec81 2c40
 0014 :  e401 24c0 2580 e541 2700 e7c1 e681 2640
 0015 :  2200 e2c1 e381 2340 e101 21c0 2080 e041
 0016 :  a001 60c0 6180 a141 6300 a3c1 a281 6240
 0017 :  6600 a6c1 a781 6740 a501 65c0 6480 a441
 0018 :  6c00 acc1 ad81 6d40 af01 6fc0 6e80 ae41
 0019 :  aa01 6ac0 6b80 ab41 6900 a9c1 a881 6840
 0020 :  7800 b8c1 b981 7940 bb01 7bc0 7a80 ba41
 0021 :  be01 7ec0 7f80 bf41 7d00 bdc1 bc81 7c40
 0022 :  b401 74c0 7580 b541 7700 b7c1 b681 7640
 0023 :  7200 b2c1 b381 7340 b101 71c0 7080 b041
 0024 :  5000 90c1 9181 5140 9301 53c0 5280 9241
 0025 :  9601 56c0 5780 9741 5500 95c1 9481 5440
 0026 :  9c01 5cc0 5d80 9d41 5f00 9fc1 9e81 5e40
 0027 :  5a00 9ac1 9b81 5b40 9901 59c0 5880 9841
 0028 :  8801 48c0 4980 8941 4b00 8bc1 8a81 4a40
 0029 :  4e00 8ec1 8f81 4f40 8d01 4dc0 4c80 8c41
 0030 :  4400 84c1 8581 4540 8701 47c0 4680 8641
 0031 :  8201 42c0 4380 8341 4100 81c1 8081 4040
```

#### DISPLAY THE DEFAULT TABLE FOR THE CCITT POLYNOMIAL 0X1021

```
dotnet bin/Debug/netcoreapp2.2/crc.dll -t --polynomial 0x1021
 0000 :  0000 1021 2042 3063 4084 50a5 60c6 70e7
 0001 :  8108 9129 a14a b16b c18c d1ad e1ce f1ef
 0002 :  1231 0210 3273 2252 52b5 4294 72f7 62d6
 0003 :  9339 8318 b37b a35a d3bd c39c f3ff e3de
 0004 :  2462 3443 0420 1401 64e6 74c7 44a4 5485
 0005 :  a56a b54b 8528 9509 e5ee f5cf c5ac d58d
 0006 :  3653 2672 1611 0630 76d7 66f6 5695 46b4
 0007 :  b75b a77a 9719 8738 f7df e7fe d79d c7bc
 0008 :  48c4 58e5 6886 78a7 0840 1861 2802 3823
 0009 :  c9cc d9ed e98e f9af 8948 9969 a90a b92b
 0010 :  5af5 4ad4 7ab7 6a96 1a71 0a50 3a33 2a12
 0011 :  dbfd cbdc fbbf eb9e 9b79 8b58 bb3b ab1a
 0012 :  6ca6 7c87 4ce4 5cc5 2c22 3c03 0c60 1c41
```

```
0013 :  edae fd8f cdec ddcd ad2a bd0b 8d68 9d49
0014 :  7e97 6eb6 5ed5 4ef4 3e13 2e32 1e51 0e70
0015 :  ff9f efbe dfdd cffc bf1b af3a 9f59 8f78
0016 :  9188 81a9 b1ca a1eb d10c c12d f14e e16f
0017 :  1080 00a1 30c2 20e3 5004 4025 7046 6067
0018 :  83b9 9398 a3fb b3da c33d d31c e37f f35e
0019 :  02b1 1290 22f3 32d2 4235 5214 6277 7256
0020 :  b5ea a5cb 95a8 8589 f56e e54f d52c c50d
0021 :  34e2 24c3 14a0 0481 7466 6447 5424 4405
0022 :  a7db b7fa 8799 97b8 e75f f77e c71d d73c
0023 :  26d3 36f2 0691 16b0 6657 7676 4615 5634
0024 :  d94c c96d f90e e92f 99c8 89e9 b98a a9ab
0025 :  5844 4865 7806 6827 18c0 08e1 3882 28a3
0026 :  cb7d db5c eb3f fb1e 8bf9 9bd8 abbb bb9a
0027 :  4a75 5a54 6a37 7a16 0af1 1ad0 2ab3 3a92
0028 :  fd2e ed0f dd6c cd4d bdaa ad8b 9de8 8dc9
0029 :  7c26 6c07 5c64 4c45 3ca2 2c83 1ce0 0cc1
0030 :  ef1f ff3e cf5d df7c af9b bfba 8fd9 9ff8
0031 :  6e17 7e36 4e55 5e74 2e93 3eb2 0ed1 1ef0
```

**CRC OF A STRING**

```
dotnet bin/Debug/netcoreapp2.2/crc.dll --polynomial 0x1021
abcdefghijklmnopqrstuvwxyz
9d86
```